HOWTO

Build and run on POSIX-compatible systems

依赖：make，gcc，gtk+-3.0，libao，libmpg123，pkg-config；

命令：make build; make run;

Build and run on Windows

环境：msys2；

依赖：make，gcc，gtk+-3.0，pkg-config，(libao，libmpg123)*;

命令：make build_win; make run_win;

验收

Windows 环境下不是最终效果，如果老师或学长是用 Windows，可以查看我上传的演示视频获得最终效果。

视频目录

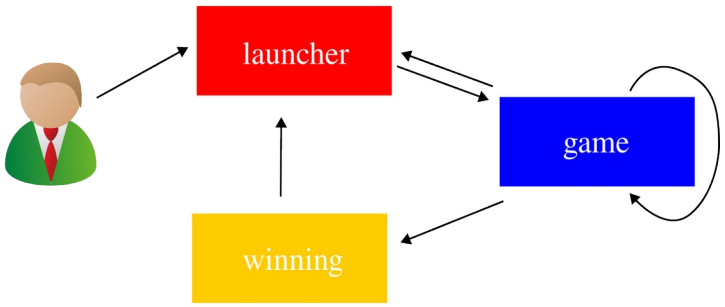
*说明：Windows 下音频播放模块因为 libmpg123 库我还没有移植成功，暂时不可用（没有背景音乐）。

Caterpillar FlightChess V3 项目汇报

一、架构

1. 程序模块

程序分为三个模块：launcher, game 和 winning。

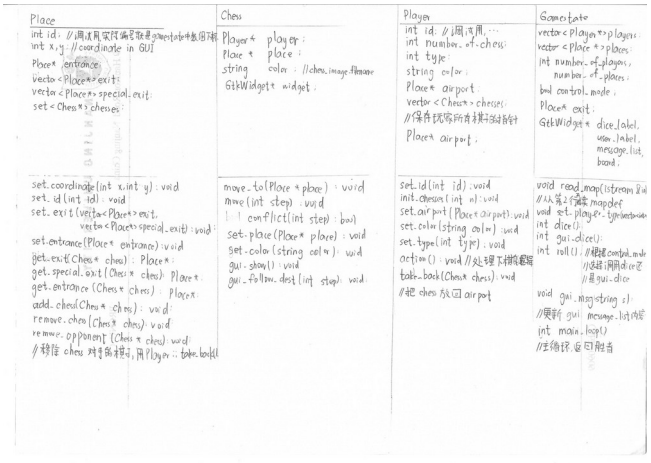


图示 1： 程序模块

其中，用户启动 launcher，launcher 检查并根据用户需要修改配置文件，然后启动 game，进入正式的游戏环节。game 结束后启动 winning，winning 宣布获胜者，被用户关闭后回到 launcher。在 game 中也可以回到 launcher 或者重启 game。

2. 设计模式

面向对象的设计模式，模块化的代码组织方式。



图示 2： 对象

面向对象设计主要体现在 game 模块中，游戏的主要数据结构是 Place, Chess, Player。棋盘的链表表示法灵感来自于隔壁 SICP proj03。

模块化的代码组织方式主要体现在音频模块。mp3player.hpp 中的 Mp3Player 类封装了用 libao 和 libmpg123 库实现的简单的音乐播放器，在三个程序模块中都有使用。

二、功能

1. 动态加载的地图。程序从文件中动态载入棋盘的描述、背景、棋子的样式等，使能够在不修改程序的条件下增加新地图、设计新规则，依靠此功能完成了拓展设计，可以载入作业要求的地图和原版地图，也可以后续加入新地图。设计的地图格式参见 map/ctpchessmap-def.conf。

2. 图形界面。依靠此功能完成了拓展设计，如图所示，用 gtk 实现了简陋的图形界面。

图示 3： GUI

3. 背景音乐。如演示视频所示，实现了简单的背景音乐播放。

4. 人工智障。时间原因，AI 暂未实现。电脑玩家只选择可移动的第一个棋子。

三、完成项目中解决的问题/学到的知识

1. 多线程线程同步和图形界面串行访问的问题，同步/异步处理

在老师和助教的帮助下，学习了 std::mutex 和 semaphore.h，并实现了符合程序逻辑的 C_SIMPLE_SIGNAL 处理函数族。

2. 图形库 GTK，MP3 解码库 libmpg123，音频输出库 libao 的使用

在学习库的使用过程中，体会到软件开发全流程中各种坑和各种“技术”的真实含义。

还有很多…

四、程序的优点

1. 通用性。程序设计的结构使得该程序能够加载各类地图，甚至稍加修改，就能变成地图类型相似的游戏，如植物大战僵尸（SICP proj03 Ants VS Somebees 😇）。
2. 可移植性。程序使用了 STL，GTK，libmpg123，libao 等跨平台可移植的库，使得程序本身可移植。
3. 代码易于理解。

五、程序的已知缺陷、不足

1. 起点的设计。程序中起点只设置了一个点，导致在机场的棋子重叠在一起。
2. 终点的设计。程序中终点处棋子相互堆叠，只显示最后加入棋盘的棋子，不知道对于每一个玩家有几个棋子到达。（以上问题可以解决，只是写了整整 2 周，写不动了）
3. 因为先实现了主要逻辑，后加了图形界面，然而 GTK 的要求是所有图形界面操作在主 thread 中完成，而我的程序在另一个线程运行，需要将后续的操作串行化，导致图形界面部分的代码不太优雅，有时打破抽象。

六、总结

第一次写小型项目，总体方向是对的，先设计架构，定义模块，然后定义接口，最后具体实现。但是设计架构时过于自信，目标太高，而具体实现中又有许多细节问题需要处理，踩了不少坑，但也收获了相应的经验，是一次有益的尝试。

七、COPYRIGHT